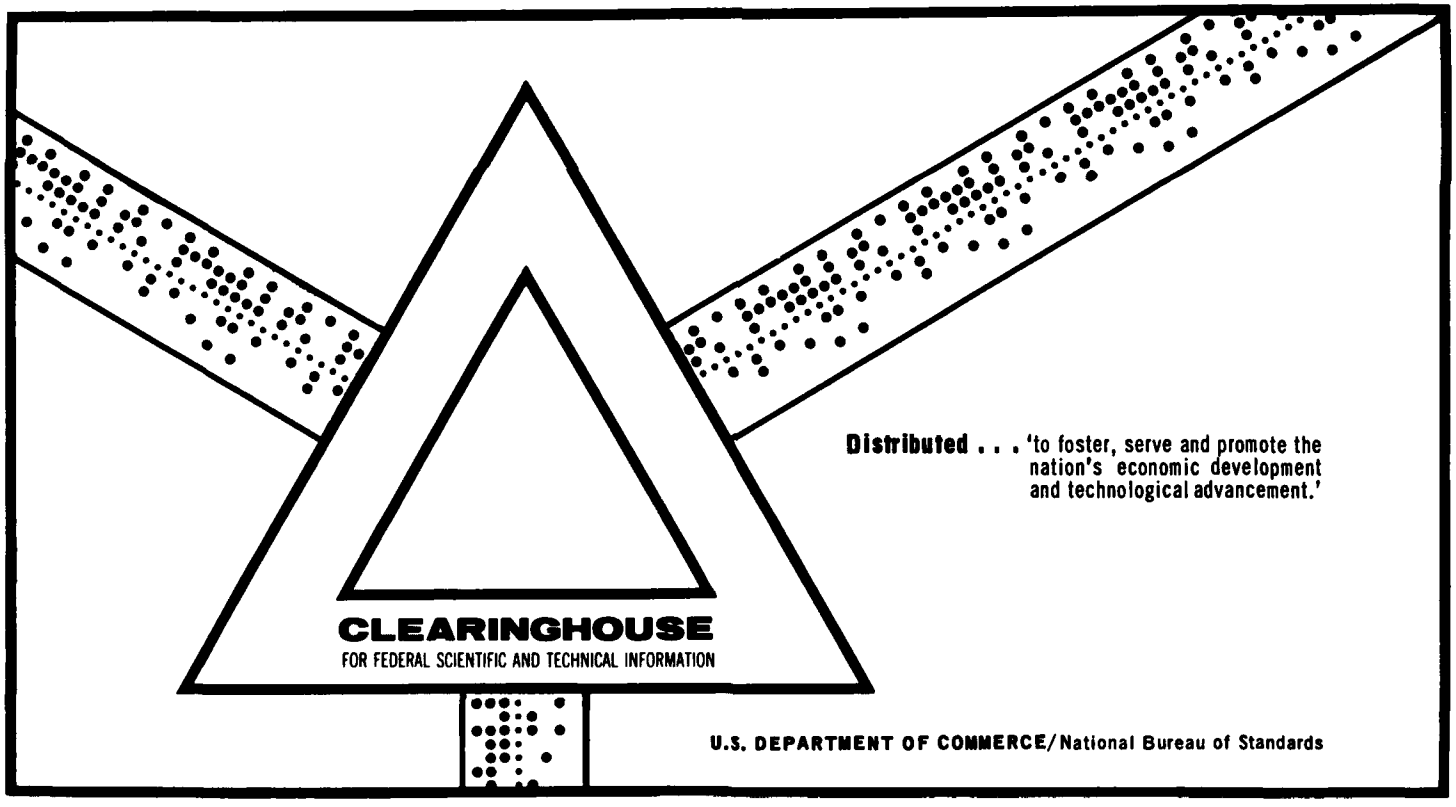
CLUSTER ANALYSIS AND MATHEMATICAL PROGRAMMING

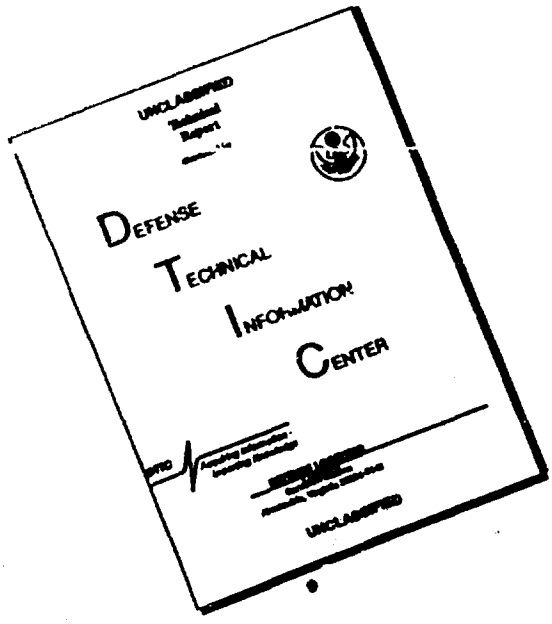
M. R. Rao

Carnegie-Mellon University
Pittsburgh, Pennsylvania

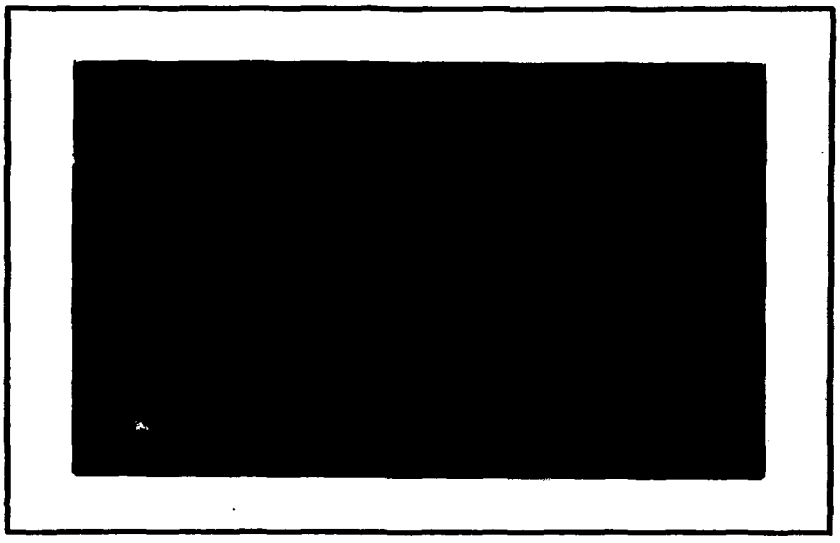
October 1969



DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

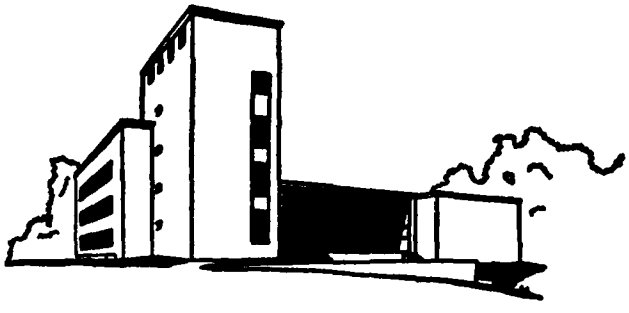


Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



Management Sciences Research Report No. 183

CLUSTER ANALYSIS AND MATHEMATICAL PROGRAMMING

by

M. R. Rao*

October 1969

* Graduate School of Industrial Administration
Carnegie-Mellon University
and
International Center for Management Sciences
at the Center for Operations Research & Econometrics
Universite Catholique de Louvain
Now at University of Rochester

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract NONR 760(24), NR 047-048 with the U. S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ACKNOWLEDGEMENT

The author is indebted to Professor W. W. Cooper for many helpful discussions. The financial support provided by the Graduate School of Industrial Administration at Carnegie Mellon University and the International Center for Management Sciences at the Center for Operations Research & Econometrics affiliated with Universite Catholique de Louvain is gratefully acknowledged.

ABSTRACT

Cluster analysis involves the problem of optimal partitioning of a given set of entities into a pre-assigned number of mutually exclusive and exhaustive clusters. Here the problem is formulated in two different ways with the distance function (a) of minimizing the within groups sums of squares and (b) minimizing the maximum distance within groups. These lead to different kinds of linear and non-linear (0-1) integer programming problems. Computational difficulties are discussed.

CLUSTER ANALYSIS AND MATHEMATICAL PROGRAMMING

M.R.RAO

INTRODUCTION

Cluster analysis involves the problem of optimal partitioning of a given set of entities into a number (pre-assigned) of mutually exclusive and exhaustive clusters. The criterion for optimality depends heavily upon the application in which it is to be used. It is not the purpose of this paper to discuss the relative merits of the various criteria. A discussion of this can be found in Sokal and Sneath [1].

In our analysis here, we confine ourselves to distance based cluster analysis in which a distance measure between the various entities is available. This type of cluster analysis has received increasing attention recently. A detailed discussion of this is given by Majone [2] and Majone and Sandy [3][1]. Even though a distance measure between the various entities is known, the criterion for optimal partitioning still depends upon the intended application. Again, it is beyond the scope of this paper to discuss the relative merits of the various possible criteria but references can be found in Jensen [5] where a dynamic programming algorithm is given for minimizing the within - groups sums of squares.

One of the purposes of this paper is to consider the criterion of minimizing the within groups sums of squares and show how the problem can be formulated as a mathematical programming problem. In the general case, the formulation leads to a fractional non-linear

---

[1] See also [4]for an application of distance based cluster analysis

0-1 programming problem with constraints and there does not appear to be any computationally efficient procedure for solving such a problem. Fortunately, the problem does appear to be computationally tractable in some variants and special cases which are discussed in detail.

An alternate criterion, viz., minimize the maximum distance within groups, is also considered. The formulation in this case leads to a linear integer (0-1) programming problem which can be solved by any one of the known techniques [6,7]. A simple but efficient algorithm is given to solve this problem when the number of pre-assigned groups is restricted to be only two.

FORMULATION OF THE PROBLEM.

The following definitions are used in the formulation :
d_{ij} is a metric distance between entities i and j.
N is the number of entities
M is the number of pre-assigned groups.
We give a formulation of the problem under different criteria.

I) Minimize (maximize) the within (between)-groups sums of squares :
Let X_{ik} be equal to 1 or 0 depending upon whether the i^{th} entity is in group k or not.
Let d_{ij} be a Euclidean metric.

The problem can now be written as

$$\text{Min} \sum_{k=1}^{M} \left[\left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 \, x_{ik} \, x_{jk} \right) \Big/ \sum_{i=1}^{N} x_{ik} \right] \qquad (1)$$

$$\text{Subject to} \sum_{k=1}^{M} x_{ik} = 1 \qquad \text{for } i = 1,2,\ldots N \qquad (2)$$

$$x_{ik} \geq 0 \text{ and integer valued for } i = 1,2,\ldots N, \quad k = 1,2,\ldots M$$

This is a fractional non-linear 0-1 programming problem which is very difficult to solve in general. However two special cases of this appear to be less difficult and they are discussed next.

a) The number of entities in each group is specified :

Sometimes the number of entities in each group is specified in advance. Let n_k be the number of entities in group k such that $\sum_{k=1}^{M} n_k = N$.

The objective function (1) now becomes

$$\text{Min} \sum_{k=1}^{M} \frac{1}{n_k} \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 \, x_{ik} \, x_{jk} \right) \qquad (3)$$

and we have an additional set of constraints, $\sum_{i=1}^{N} x_{ik} = n_k$

$$k = 1,2,\ldots M \qquad (4)$$

This is still a non-linear 0-1 programming problem with constraint set (2) and (4). However since all n_k are specified in advance, we do not have a fractional objective as in (1) and there are at least two possible approaches to attempt to solve this problem. The first approach is to treat this as a constrained non-linear boolean programming problem and use the methods outlined by Hammer and Rudeanu [8]. Another approach is to linearize the objective function at the expense of increasing the number of constraints and solve the resulting problem by any of the known techniques for linear integer programming. The 0-1 programming problem to be solved is as follows :

$$\text{Min} \sum_{k=1}^{M} \frac{1}{n_k} \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 \, y_{ij}^k \right) \qquad (5)$$

$$\text{Subject to } x_{ik} + x_{jk} - y_{ij}^k \leq 1 \qquad \begin{matrix} i = 1,2,\ldots N-1 \\ j = i+1, i+2, \ldots N \\ k = 1,2,\ldots M \end{matrix} \qquad (6)$$

$$\sum_{i=1}^{N} x_{ik} = n_k \qquad k = 1,2,\ldots M$$

$$\sum_{k=1}^{M} x_{ik} = 1 \qquad i = 1,2,\ldots N$$

All x_{ik} and $y_{ij}^{k} \geq 0$ and integer valued.

In this formulation, the number of constraints increases rapidly with N and M and hence this approach is computationally useful only for small values of N and M.

b) The number of pre-assigned groups is equal to two :
A method for cluster analysis suggested by Edwards and Cavalli-Sforza [9] involves dividing the entities into two most-compact clusters and repeating the process sequentially. This, in our notation, implies that at each stage of the process we have M = 2.

In this case, a better formulation of the problem is possible with the following notation :
Let x_i = 1 or 0 depending upon whether the i^{th} entity is in group 1 or 2.
The problem can be written as

$$\text{Min}\left[\left(\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}^2\, x_i\, x_j\right)\Big/\sum_{i=1}^{N} x_i\right]+\left[\left\{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}^2(1-x_i)(1-x_j)\right\}\Big/\left(N-\sum_{i=1}^{N} x_i\right)\right] \quad (7)$$

$$x_i = 0 \text{ or } 1 \text{ for all } i.$$

This is a fractional non-linear 0-1 programming problem with no

additional constraints. This is also a boolean programming problem an one approach to solve this problem is to use the methods given in [8]. We outline two other methods and of these the second one appears to be promising. It should be pointed out however, that so far no computational experience is available.

1) In the first approach, we rewrite the objective function (7) as follows :

$$\text{Min } [(N - \sum_{i=1}^{N} x_i)(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 x_i x_j) + \{\sum_{i=1}^{N} x_i\}\{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}^2 (1-x_i)(1-x_j)\}]/(\sum_{i=1}^{N} x_i)(N - \sum_{i=1}^{N} x_i) \quad (8$$

Let the objective function (8) without the denominator be referred to as (8'). We note that the denominator in (8) is a maximum if

$$\sum_{i=1}^{N} x_i = [\tfrac{N}{2}] \quad \text{where}$$

$[\frac{N}{2}]$ is the least integer greater than or equal to $N/2$.

We first minimize the objective function (8') by using the methods given in [8]. If we are fortunate to obtain

$$\sum_{i=1}^{N} x_i = [\tfrac{N}{2}] \quad \text{or} \quad [\tfrac{N}{2}] - 1$$

the problem is then solved. Otherwise, let $\sum_{i=1}^{N} x_i = m < [\frac{N}{2}]$.*

Now, we know that the only possible solutions that may improve the value of (8) should necessarily satisfy the following relation :

$$m < \sum_{i=1}^{N} x_i < N - m \quad (9)$$

---

* There is no loss in generality in assuming that $m < [\frac{N}{2}]$ since if this were not true we could use $N - \sum_{i=1}^{N} x_i$ instead of $\sum_{i=1}^{N} x_i$.

The next step is to consider (8') and solve the problem with the added constraint (9). If the current solution has a better value for the objective function (8) than the best value found till now, it is retained as the best value found thus far. The process is repeated until either $\sum_{i=1}^{N} x_i = [\frac{N}{2}]$ or the value of the function (8') becomes so high that the value for the objective function (8) would be higher than the current best value even if we assume $\sum_{i=1}^{N} x_i$ to be equal to $[\frac{N}{2}]$ or $[\frac{N}{2}] - 1$.

The efficiency of this approach depends heavily upon the ability to find the solution of (8') with the restriction (9). At present, there does not appear to be efficient methods for doing this.

2) An alternate approach which appears to be promising is to use branch and bound methods to solve this problem. First we note that the problem is very similar to the problem of "selecting an optimum subset" described by Beale [10] where a tree-search (branch and bound) procedure is outlined. Using the terminology in [10], we briefly describe the procedure first and then indicate in some detail how the various steps can be performed for this problem.

The root of the tree corresponds to a partial solution in which all the elements[2] are free to be assigned. We select one element and we now have two possibilities of assignment. We can assign the element to group 1 or group 2. These two possibilities correspond to the two branches emanating from the root. For convenience, we will always write the assignment of an element to group 1 as the branch to the right and always branch to the right

---

[2] For convience we use the word element to refer to an entity.

first. Thus, at any node of the tree, we have a partial solution with some elements assigned to group 1, some assigned to group 2 while others are yet to be assigned. A complete solution is one in which all the elements have been assigned and this corresponds to an end of the tree.

Before we branch from a particular node, we compute a lower bound for the objective function. If the lower bound is greater than or equal to the best complete solution found so far, we do not branch further from this node. In this case, we back-track along the tree until we reach a node with an unexplored branch to the left. If no such node exists, the problem is solved and the best complete solution found so far is the optimal solution. In the other case, where the lower bound is less than the value of the current best solution, we need to branch further. We select a variable not yet assigned and branch to the right. We continue branching until we reach an end of the tree or the lower bound test indicates that we need not branch further. When an end of the tree is reached, it would represent an improvement in the objective function and hence it is retained as the current best solution. From the end of the tree, we then back-track along the tree until we reach a node with an unexplored branch to the left.

SELECTION OF AN ELEMENT AT A NODE.

At any particular node, we have a set (possibly empty) of elements in group 1 and a set (possibly empty) of elements in group 2. If we need to branch further, we select an element not yet assigned and branch to the right. In our procedure, this corresponds to assigning this selected element to group 1. So the selection of an element is made such that the sum of the distances between this element and all other elements already in group 1 is minimum.

COMPUTATION OF LOWER BOUND AT A NODE.

We first give some notations for convenience. At a given node let

$x_i = 1$ for all i assigned to group 1

$x_i = 0$ for all i assigned to group 2

Let $N_1 = \{i/x_i = 1\}$, and $N_2 = \{i/x_i = 0\}$.

Let n_1 and n_2 be the number of elements in N_1 and N_2 respectively.

Let $n_3 = N - n_1 - n_2$ be the number of elements not yet assigned to any group and let N_3 be a set consisting of these n_3 elements.

Let $K_1 = \sum_{i,j \in N_1} d_{ij}$ and $K_2 = \sum_{i,j \in N_2} d_{ij}$

Let D be a N x N symmetric matrix giving the distance between the entities. Let $\mathbb{F}$ be a n_3 x n_1 sub-matrix of D giving the distance between an element in N_3 and an element in N_1. Let $\mathbb{F}_i$ be the sum of the elements in row i of matrix $\mathbb{F}$.

Let $\mathbb{H}$ be a n_2 x n_1 submatrix of D giving the distance between an element in N_2 and an element in N_1. Let H_i be the sum of the elements in row i of the matrix $\mathbb{H}$.

We assume that matrices $\mathbb{H}$ and $\mathbb{F}$ are arranged in such a manner that

$$\mathbb{H}_i \leq \mathbb{H}_j \qquad \text{for } i < j.$$

$$\mathbb{F}_i \leq \mathbb{F}_j \qquad \text{for } i < j.$$

Let C be a n_3 x n_3 symmetric sub-matrix of D giving the distance between each pair of elements in N_3. The diagonal elements of C are assigned a very high value (∞).

We are interested in finding

$$Z = \text{Min}\left\{\frac{K_1 + \sum_{i\in N_1, j\in N_3} d_{ij}\, x_i\, x_j + \sum_{i,j\in N_3} d_{ij}\, x_i\, x_j}{n_1 + \sum_{i\in N_3} x_i} + \frac{K_2 + \sum_{i\in N_2, j\in N_3} d_{ij}(1-x_i)(1-x_j) + \sum_{i,j\in N_3} d_{ij}(1-x_i)(1-x_j)}{n_2 + (n_3 - \sum_{i\in N_3} x_i)}\right. \quad (10)$$

where $x_i = 0$ or 1

For any fixed value for $\sum_{i\in N_3} x_i = n_3'$, let $n_3'' = n_3 - n_3'$, $p = n_1 + n_3'$ and $t = n_2 + n_3''$. Now we can write (10) as

$$Z' = \text{Min}\; t\Big[\sum_{i\in N_1, j\in N_3} d_{ij}\, x_i\, x_j + \sum_{i,j\in N_3} d_{ij}\, x_i\, x_j\Big] + p\Big[\sum_{i\in N_2, j\in N_3} d_{ij}\,(1-x_i)(1-x_j) + \sum_{i,j\in N_3} d_{ij}(1-x_i)(1-x$$

Since $K_1 t$ and $K_2 p$ are constants and the denominator is also a constant equal to pt.

Now, there are several ways of finding a lower bound for Z' and some of these are listed below.
By varying n_3' over its range 0 to n_3, we obtain $n_3 + 1$ lower bounds for Z' and the least of the lower bounds gives the desired bound for Z in (10). We should point out that although it might at first appear to be time consuming to get $n_3 + 1$ lower bounds for Z', many of these calculations are very easy to perform. Of course, without computational experience, it is difficult to say how effective would be the bound for Z obtained in this manner.

i) $Z' \geq u_1 + u_2 + u_3$

where $u_1 = (t)\ \text{Min} \sum_{i\in N_1, j\in N_3} d_{ij}\, x_i\, x_j = (t) \sum_{i=1}^{n_3'} F_i$

$u_2 = (p)\ \text{Min} \sum_{i\in N_2, j\in N_3} d_{ij}(1-x_i)(1-x_j) = (p) \sum_{i=1}^{n_3''} H_i$

$u_3 = [(t)\text{Min} \sum_{i,j\in N_3} d_{ij}\, x_i\, x_j] + [(p)\ \text{Min} \sum_{i,j\in N_3} d_{ij}(1-x_i)(1-x_j)]$

In order to find u_3 we first note that the summation (Σ) terms in u_3 consist of $v = [n_3'(n_3'-1)+n_3''(n_3''-1)]/2$ terms in either the upper[3] or the lower triangular part of the symmetric matrix C. Therfore a lower bound for u_3 is given by Min [t, p] multiplied by the sum of the v smallest elements of the upper half of matrix C.

ii) $Z' \geq u_1 + u_2 + u_4$

where u_1 and u_2 are as given in i) and

$u_4 = \text{Min}\ [t(\sum_{i,j\in N_3} d_{ij}\, x_i\, x_j) + p\{\sum_{i,j\in N_3} d_{ij}(1-x_i)(1-x_j)\}]$

Let[4] $t \leq p$.

$u_4 = \text{Min}\ [t(\sum_{i,j\in N_3} d_{ij}\{x_i\, x_j + (1-x_i)(1-x_j)\}) + (p-t) \sum_{i,j\in N_3} d_{ij}(1-x_i)(1-x_j)$

$\geq u_5 + u_6$

where $u_5 = t\ \text{Min}\ [\sum_{i,j\in N_3} d_{ij}\{x_i\, x_j + (1-x_i)(1-x_j)\}]$

---

[3] We will take the upper triangular part of the matrix C.

[4] If t > p, a similar procedure would hold but we do not repeat the details here.

which is equal to t multiplied by the sum of the v terms of the matrix C (as described in i.)

$$u_6 = (p-t)\ \text{Min} \sum_{i,j \in N_3} d_{ij}(1-x_i)(1-x_j)$$

In order to find u_6 we note that the summation term consists of $r = n_3''(n_3''-1)/2$ terms in the upper half matrix C. So a lower bound for u_6 is given by (p-t) multiplied by the r smallest elements of the upper half of C.

iii) $Z' \geq u_7 + (u_3 \text{ or } u_4)$

where u_3 and u_4 are as given in i) and ii) respectively and

$$u_7 = \text{Min}\ [t(\sum_{i \in N_1, j \in N_3} d_{ij}\ x_i\ x_j) + p\{\sum_{i \in N_2, j \in N_3} d_{ij}\ (1-x_i)(1-x_j)\}]$$

In i) we used a lower bound of $u_1 + u_2$ for u_7. A better lower bound can be obtained for u_7 by solving a transportation problem [11] as indicated below. Let the elements of N_3 be numbered as $1,2,\ldots n_3$. Let F^i be the sum of all the elements of a row in F corresponding to element i in N_3. Similarly, let H^i be the sum of all the elements of a row in H corresponding to element i in N_3.

Now, we have to subdivide the n_3 elements of N_3 into two subsets one consisting of n_3' elements and the other consisting of n_3'' elements. The two divisions can be considered as two demand points, the first one requiring n_3' units and the other requiring n_3'' units. The n_3 elements can be considered as n_3 supply points each with a supply of one unit. The cost of transporting a unit

from the i^{th} supply point to the first and second demand points is given by $t\,B^i$ and $p\,A^i$ respectively. The minimum transportation cost gives u_7 which is a better bound than $u_1 + u_2$. Of course, in order to obtain this better bound, we have to solve a transportation problem which can be time consuming since it has to be repeated $n_3 + 1$ times to obtain a bound for Z. But, some of these (for e.g. $n_3' = 0$ or $n_3' = n_3$ etc.) are trivial calculations and do not require a transportation problem to be solved.

The efficiency of the branch and bound procedure outlined here depends upon the effectiveness of the lower bound at a node which is difficult to predict without computational experience. An additional feature described in [10] is the restructuring of the tree and this can be incorporated here also. However, here we omit the details which are given in [10].

II. Minimize the total within group distance.

If we are interested in only minimizing the total within groups distance[5], the objective function (3) would be modified so that the constant term $1/n_k$ does not appear and the term d_{ij}^2 is replaced by d_{ij}. As before, the constraint set is given by (2). If necessary, by adding the constraint (6) we can

---

[5] If the total between groups distance is defined to be the sum of all the distances between entities that do not belong to the same group, then minimizing the total within groups distance is equivalent to maximizing the total between groups distance. Cooper and Majone [12] suggest a criterion of minimizing (maximizing) the total within (between) groups distance subject to a constant level of between (within) cluster distances. It should be possible to obtain a constant level of between cluster distances by adding appropriate constraints.

linearize this new objective and obtain a function very similar to that given in (5).

If the number of groups (M) is restricted to be two, the objective (7) would now become

$$\text{Min } \left(\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}\, x_i\, x_j \right) + \left\{ \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} d_{ij}\, (1-x_i)(1-x_j) \right\}$$

Since the only constraints in the problem are

$$x_i = 0 \text{ or } 1 \qquad \text{for all } i$$

this problem can be solved without difficulty by the methods given in [8].

III. Minimize the maximum within group distance.

With this criterion, instead of minimizing the total within group distance, we would minimize the maximum distance within groups. A mathematical formulation of the problem is

$$\text{Min } Z$$

$$\text{Subject to } d_{ij}\, x_{ik} + d_{ij}\, x_{jk} - Z \leq d_{ij}$$

$$i = 1,2,\ldots N-1$$
$$j = i+1, i+2 \ldots N$$
$$k = 1,2,\ldots M$$

$$\sum_{k=1}^{M} x_{ik} = 1 \qquad i = 1,2,\ldots N$$

x_{ij} and $Z \geq 0$ and integer valued[6]

This is an integer linear programming problem but unfortunately the number of constraints increases very rapidly with N and M and hence this formulation is computationally useful only for small values of N and M.

It should be pointed out that if we solve this problem as a linear programming problem without the variables being restricted to integers, the solution is $x_{ik} = \frac{1}{M}$ for all i and k. Hence we would never be fortunate to obtain an integer solution as the linear programming solution. However when the number of groups (M) is equal to two, this problem can be solved efficiently as outlined below.

a) Number of groups (M) equal to two :

Let D be the matrix giving the distance d_{ij} between the entities i and j.
Let A and B refer to the two groups. If an element receives a label A (B) it means that it is assigned permanently to group A (B). A label k (integer between 1 and N) to an element is a temporary label.

An algorithm to solve the problem is given next and a flow chart is provided in figure 1.

---

[6] Without any loss in generality we may assume that all d_{ij} are positive integer valued and hence Z will also be an integer.

1. Let d_{ij} be the largest value[7] in the matrix M.
2. i receives a label A and j receives a label B.
3. Let $d_{ij} = -1$.[8] Let d_{pq} be the current largest value in M. Let i = p and j = q. If both i and j have not received any label go to 4. Otherwise go to 5.
4. Assign i a label i and j a label $\bar{i}$. Go to 3.
5. If either i or j but not both has received a label A or B go to 6. Otherwise go to 8.
6. If j has a label A or B go to 7. Otherwise i has a label A (B) but j does not have a label A or B. If j does not have a label, label j as B (A) and go to 3. If j has a label k, give a label B (A) to all elements with label k and a label A (B) to all elements with label $\bar{k}$. If j has a label $\bar{k}$, give a label B (A) to all elements with label $\bar{k}$ and a label A (B) to all elements with label k. Erase the labels k and $\bar{k}$. Go to 3.
7. j has a label A (B) but i does not have a lable A or B. If i does not have a label, label i as B (A) and go to 3. If i has a label k, give a label B (A) to all elements with label k and a label A (B) to all elements with label $\bar{k}$. If i has a label $\bar{k}$, give a label B (A) to all elements with label $\bar{k}$ and a label A (B) to all elements with label k. Erase the labels k and $\bar{k}$. Go to 3.
8. If both i and j have the same label A (B) go to 16. If i has a label A (B) and j has a label B (A) go to 3. Otherwise go to 9.

---

[7] In case of ties, for unambiguity choose the one with the smallest index i. To break ties further, if any, choose the smallest index j.

[8] We have assumed that all d_{pq} are non-negative. If some d_{pq} are negative, we would set d_{ij} equal to a large negative number.

9. Both i and j do not have a label A or B. If either i or j but not both has been labeled before, go to 10. Otherwise go to 12.
10. If j has a label $k(\bar{k})$ go to 11. Otherwise i has a label. Let the label of i be $k(\bar{k})$. Give j a label $\bar{k}(k)$. Go to 3.
11. Let the label of j be $k(\bar{k})$. Give i a label $\bar{k}(k)$. Go to 3.
12. Both i and j have a label (but neither of them has a label A or B). If both i and j have the same label $k(\bar{k})$ go to 16. Otherwise go to 13.
13. If i is labeled $k(\bar{k})$ and j is labeled $\bar{k}(k)$ go to 3. Otherwise go to 14.
14. i has a label $s(\bar{s})$. If j has a label $\bar{t}$ go to 15. Otherwise j has a label t. Assign a label $\bar{s}(s)$ to all elements with label t and assign a label $s(\bar{s})$ to all elements with label $\bar{t}$. Erase the labels t and $\bar{t}$. Go to 3.
15. Assign a label $\bar{s}(s)$ to all elements with label $\bar{t}$ and assign a label $s(\bar{s})$ to all elements with label t. Erase the labels t and $\bar{t}$. Go to 3.
16. The minimum of the maximum distance within groups is equal to d_{ij}. For k = 1,2,...N, all elements with label k are assigned a label A and all elements with label $\bar{k}$ are assigned a label B. Erase all labels k and $\bar{k}$. Remaining elements, if any, that have not been examined so far can be assigned a label A or B arbitrarily[9]. END

---

[9] We have assumed here that we are interested in only minimizing the maximum within group distance. A better approach would be to first assign some of the elements to the two groups so that the maximum within group distance is minimized. Once this is achieved, other elements may be assigned to the two groups so that the minimum distance (considering only those elements not yet assigned) between groups is maximized.

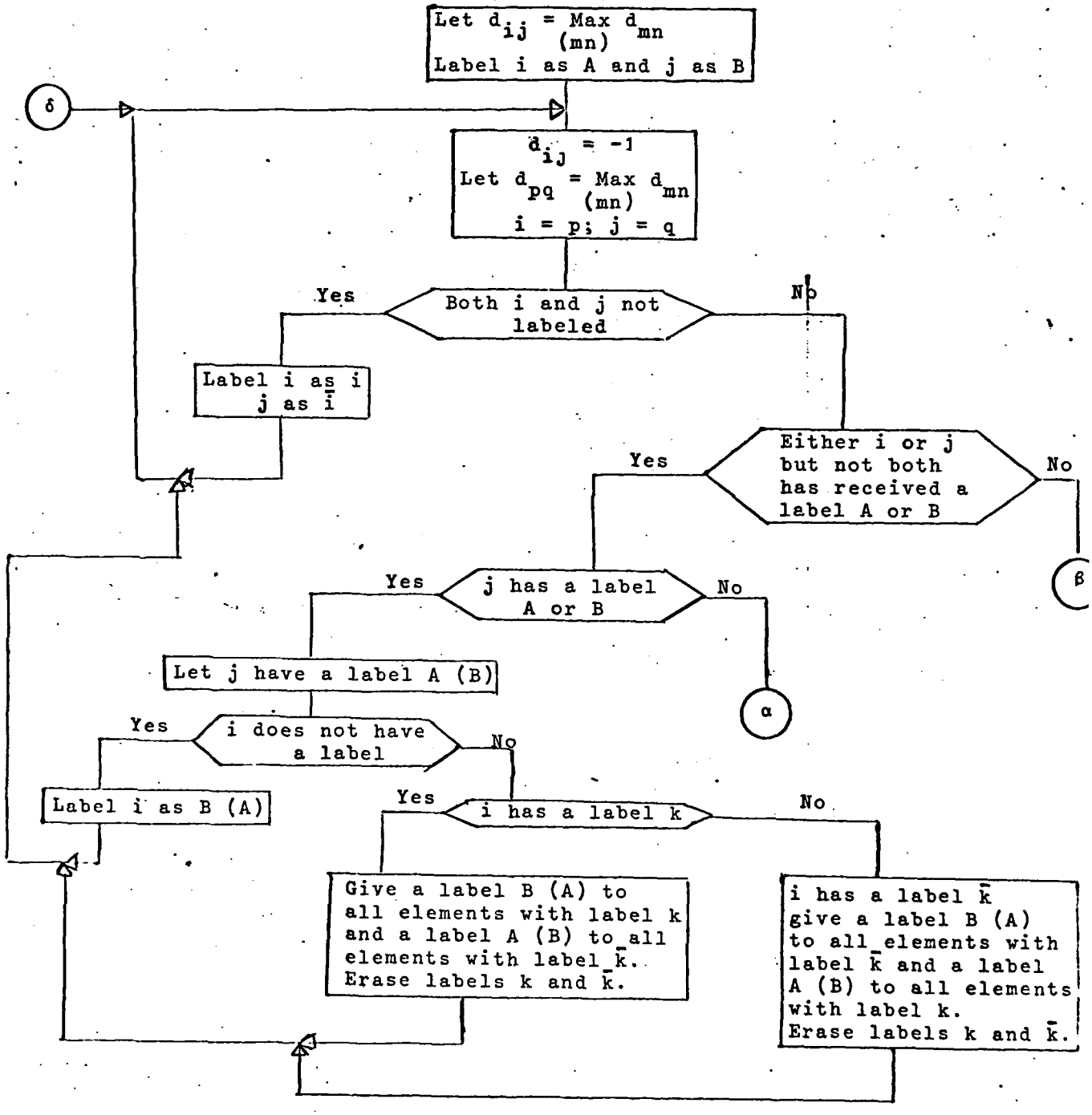
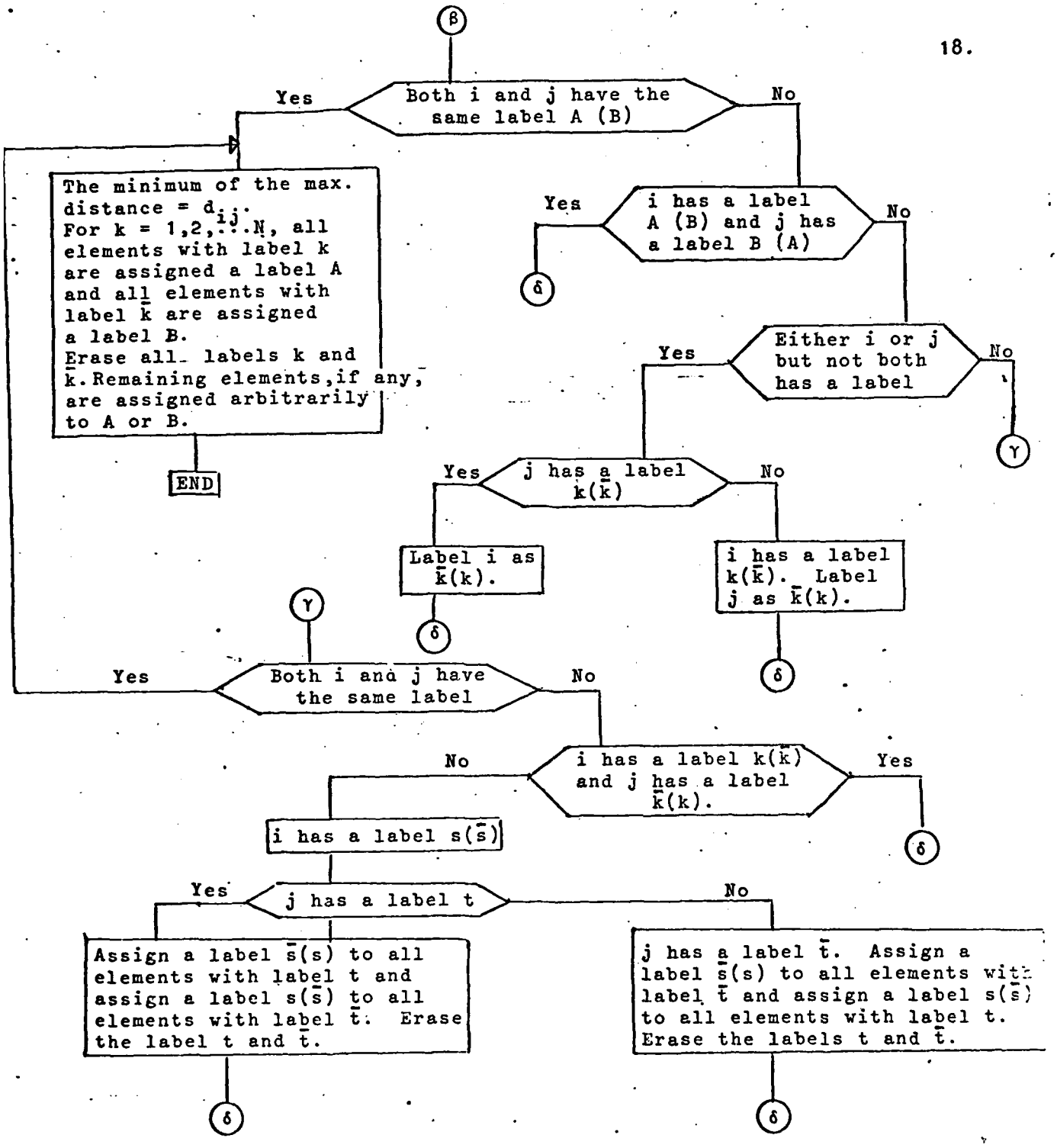
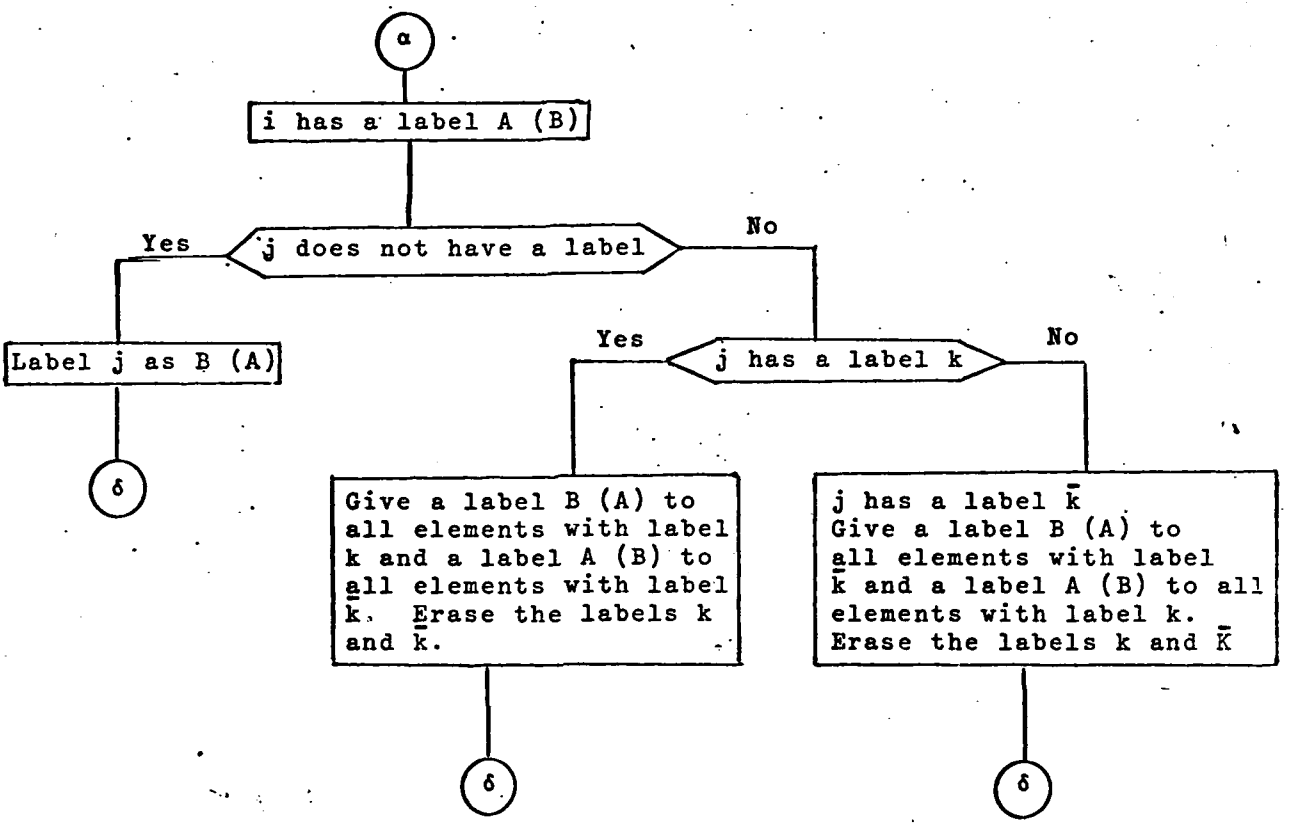


Figure 1
Flow chart for the algorithm.



Flow chart for the algorithm (contd.)



Flow chart for the algorithm (contd.)

CONCLUSIONS.

In this paper we have shown how some of the problems in distance-based cluster analysis can be viewed as a mathematical programming problem. Many of the problems are very difficult to solve and require further research in terms of solution techniques. But as we have indicated here, the problem of minimizing the maximum within group distance is easily solvable when the number of groups is restricted to two. Some of the other problems appear to be computationally tractable and deserve some computations experience before drawing further conclusions.

REFERENCES.

1. SOKAL, R.G., and P.H.A. SNEATH "Principles of Numerical Taxonomy"; W.H. Freeman and Company, San Francisco, 1963.

2. MAJONE, G. "Distance-based cluster Analysis and Measurement Scales", Working Paper no. 17, November 1968, University of British Columbia, Vancouver, B.C., Canada.

3. MAJONE, G. and P.R. SANDAY, "On the Numerical Classification of Nominal Data" Management Sciences Research Report no. 118, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

4. MAJONE, G. "Classification and Discrimination in the Analysis of Credit Risks : I", Management Sciences Research Report no. 120, February 1968, Carnegie-Mellon University, Pittsburgh, Pennsylvania.

5. JENSEN, R.E. "A Dynamic Programming Algorithm for Cluster Analysis", Working Paper no. 20, University of Maine, Orono, Maine.

6. GOMORY, R.E., " All-Integer Integer Programming Algorithm" in Muth, J.F. and G.L. Thompson (eds.), Industrial Scheduling, Prentice-Hall, Englewood Cliffs, 1963.

7. BALAS, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", Operations Research, Vol. 13, 1965, pp. 517-546.

8. HAMMER, P.L. and S. Rudeanu "Boolean Methods in Operations Research and Related Areas" Springer-Verlag 1968.

9. EDWARDS, A.W.F. and L.L. CAVALLI-SFARZA, "A Method for Cluster Analysis" Biometrics, June 1965, pp. 362-375.

10. BEALE, E.M.L., "Selecting an Optimum Subset", NATO Summer School on Integer and Nonlinear Programming", Bandol, France 1969.

11. CHARNES, A.A., and W.W. COOPER, "Management Models and Industrial Applications of Linear Programming, Vol. I, John Wiley & Sons, Inc., 1961.

12. COOPER, W.W. and G. MAJONE,"A Description and Some Suggested Extensions for Methods of Cluster Analysis", Internal Working Memorandum, Carnegie-Mellon University.

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)

Graduate School of Industrial Administration
Carnegie-Mellon University

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

Not applicable

3. REPORT TITLE

CLUSTER ANALYSIS AND MATHEMATICAL PROGRAMMING

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)

Management Sciences Research Report, October 1969

5. AUTHOR(S) (First name, middle initial, last name)

M. R. Rao

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
October 1969	24	12

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
NONR 760(24)	Management Science Research Report No. 183
b. PROJECT NO. NR 047-048	
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	Not applicable

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
Not applicable	Logistics and Mathematical Statistics Br. Office of Naval Research Washington, D. C. 20360

13. ABSTRACT

Cluster analysis involves the problem of optimal partitioning of a given set of entities into a pre-assigned number of mutually exclusive and exhaustive clusters. Here the problem is formulated in two different ways with the distance function (a) of minimizing the within groups sums of squares and (b) minimizing the maximum distance within groups. These lead to different kinds of linear and non-linear (0-1) integer programming problems. Computational difficulties are discussed.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6811

Unclassified

Security Classification

A-31408

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Cluster analysis						
Integer linear programming						
Integer non-linear programming						
Partitioning of a set						